Управление параллельными заданиями

Капустин Александр Сергеевич

28 августа 2014 г.

История

- Однопоточные
- Кооперативная многозадачность
- Вытесняющая многозадачность (изолированные процессы) (IO)
- Поток (без виртуальной памяти). НО!!! опять общая память..

Проблемы

- Проблема достоверности
- Утраченные изменения
- Несогласаованное чтение

Решение - один пользователь общается с одним типом данных
Но, страдает параллелизм...

Контексты выполнения

- Запрос (обычно без прерывания сессии- запрос с отменой)
- Сеанс(сессия)
- Транзакции

Изолированность и устойчивость

Можно использовать изолированность как у процесса или у блокировки файлов.
Если понять какие данные достаточно устойчивы, можно ослабить требование изолированности

Стратегии блокирования

- Оптимистические - как в свн, git, блокировка только при фиксации изменений
- Пессимистические -полная блокировка

Предотвращение возможности несогласованного чтения данных

Допустим, я использую класс наследника, а кто-то поменял базовый класс...

- Оптимистические - как в свн, git, блокировка только при фиксации изменений
- Пессимистические - полная блокировка

Разрешение взаимоблокировок

- Определение жертв - отдают свою блокировку другому
- Лимит времени для каждой блокировки
- Можно улучшить дело - запретить брать больше чем сколько-нибудь блокировок

ACID свойства:

- Atomicity (атомарность). В контексте транзакции либо выполняются все действия, либо не выполняется ни одно из них. Частичное или избирательное выполнение недопустимо. Например, если клиент банка переводит сумму с одного счета на другой и в момент между завершением расходной и началом приходной операции сервер терпит крах, система должна вести себя так, будто расходной операции не было вовсе. Система должна либо осуществить обе операции, либо не выполнить ни одной. Фиксация (commit) результатов служит свидетельством успешного окон чания транзакции; откат (rollback) приводит систему в состояние, в котором она пребывала до начала транзакции.

- Consistency (согласованность). Системные ресурсы должны пребывать в целостном и непротиворечивом состоянии как до начала транзакции, так и после ее окончания.
- Isolation (изолированность). Промежуточные результаты транзакции должны быть закрыты для доступа со стороны любой другой действующей транзакции до момента их фиксации. Иными словами, транзакция протекает так, будто в тот же период времени других параллельных транзакций не существует.
- Durability (устойчивость). Результат выполнения завершенной транзакции не должен быть утрачен ни при каких условиях.

Обычно системы проектируются на короткие транзакции, но бывают и длинные... Отсроченные транзакции- плохи для параллелизма и согласованности данных

Уровни изоляции:

- Упорядоченные транзакции - идут строго параллельно
- Повторяемое чтение - постоянное считывание - данные меняются во время транзакции
- Чтение фиксированных данных - что есть до начала с тем и работаем
- Чтение нефиксированных данных - можем заглянуть и в данные, которые на данный момент меняются

Типовые решения задачи обеспечения автономного параллелизма

- Оптимистическая автономная блокировка
- Пессимистическая автономная блокировка

- Пулл процессов - важно, чтоб ресурс корректно и своевременно освобождался
- Поток на запрос - более эффективен с точки зрения ос, но не изолирован

Спасибо за внимание!!!